SweetWiki: a semantic wiki

Michel Buffa[1], Fabien Gandon[2], Guillaume Ereteo[2], Peter Sander[1], Catherine Faron[1]

[1]KEWI Group, I3S Laboratory,
University of Nice, France
buffa@unice.fr, (sander,
faron)@polytech.unice.fr

[2]Edelweiss Group, INRIA
Sophia-Antipolis, France
(Guillaume.Ereteo,
Fabien.Gandon@sophia.inria.fr)

ABSTRACT

Everyone agrees that user interactions and social networks are among the cornerstones of "Web 2.0". Web 2.0 applications generally run in a web browser, propose dynamic content with rich user interfaces, offer means to easily add or edit content of the web site they belong to and present social network aspects. Well-known applications that have helped spread Web 2.0 are blogs, wikis, and image/video sharing sites; they have dramatically increased sharing and participation among web users. It is possible to build knowledge using tools that can help analyze users' behavior behind the scenes: what they do, what they know, what they want. Tools that help share this knowledge across a network, and that can reason on that knowledge, will lead to users who can better use the knowledge available, i.e., to smarter users. Wikipedia, a wildly successful example of web technology, has helped knowledge-sharing between people by letting individuals freely create and modify its content. But Wikipedia is designed for people - today's software cannot understand and reason on Wikipedia's content. In parallel, the "semantic web", a set of technologies that help knowledge-sharing across the web between different applications, is starting to gain attraction. Researchers have only recently started working on the concept of a "semantic wiki", mixing the advantages of the wiki and the technologies of the semantic web. In this paper we will present a state-of-the-art of semantic wikis, and we will introduce SweetWiki, an example of an application reconciling two trends of the future web: a semantically-augmented web and a web of social applications where every user is an active provider as well as a consumer of information. SweetWiki makes heavy use of semantic web concepts and languages, and demonstrates how the use of such paradigms can improve navigation, search, and usability.

Categories and Subject Descriptors

K.4.3 [**Organizational Impacts**]: Computer-supported collaborative work.

General Terms

Management, Measurement, Documentation, Experimentation, Human Factors, Standardization.

Keywords

Wiki, Semantic Web, Social Tagging, Ontology, Web 2.0.

1. Introduction

The wiki revolution started in 1995 when Ward Cunningham wrote the first wiki for the Portland Pattern Repository[1]. Tired of centralized web publishing, of the complexity of HTML page production, and influenced by Hypercard and by the initial vision of the Web, he created a web site where people could create, modify, refactor and link pages all from within their web browser, in a very simple - one click - way. Instead of HTML he proposed a stripped-down markup language (WikiML) inspired by the way people were formatting text-only messages in the days before Internet was multimedia. Cunningham's biggest contribution from our point of view is the invention of WikiWords as a means to create hyperlinks[2], even to pages not yet created. Type a WikiWord (e.g. NewPage) and it will be saved as a link to a page whose URL ends with this WikiWord. If the page does not exist, clicking on the link creates it. The word "wiki" means "quick" in the Hawaiian language and it was all about quick and easy ways to create and edit web sites. Non-technical people could handle it and wikis started to grow exponentially. Nowadays, perhaps the most famous example of a public wiki is the Wikipedia.

However, studies have shown that the acceptance of such open, low-structured collaborative tools is not automatic for most intranets or community sites. There are different reasons for lower than might be expected acceptance, including social reasons, e.g., corporate culture may not be adapted, as well as usability reasons, e.g., the wiki is not structured enough, it is hard to navigate and to find relevant information, the wiki markup language used by most wiki engines makes people reluctant to contribute to the wiki, etc. (see [8], [11] and [37]). Most wiki engines, the software behind wiki sites, were designed in the mid-nineties exploiting the web technologies of the time, i.e., mainly HTML, HTTP, CGI and URIs. Inevitably, wikis developed markup languages which were variants of WikiML, and there is now no standard WikiML in spite of recent standardization efforts, including the CREOLE project [17] and the Wiki Interchange Format (WIF) project [50]).

A semantic wiki is a wiki engine that uses technologies from the semantic Web[3] to embed formalized knowledge, content, structures and links, in the wiki pages. Formalized knowledge is represented using semantic web frameworks and is thus accessible

[1] http://www.c2.com, still active. This site has entered the Web Hall of Fame now and is commonly cited as "Ward's wiki".

[2] This syntax is inspired by the naming conventions of the SmallTalk language classes. SmallTalk is itself a WikiWord.

[3] http://www.w3.org/2001/sw/

and reusable by web applications. Within the wiki, this knowledge can be used to propose enhanced features such as better document searching, suggesting new links, identifying acquaintance networks, dynamic content update, checking and notification, etc.

Current semantic wikis are either built on top of existing regular wiki engines and propose semantic web extensions, or have been created from scratch with semantic web technologies in mind. Some wikis are dedicated to editing ontologies cooperatively, others use ontologies as a reference for annotating wiki content, and some do both. Some wikis use specialized editors for the semantic content, and some use markups for adding semantic meaning. Some semantic wikis embed a reasoning engine, some can export the annotations or the ontologies defined in the wiki as RDF or RDFS/OWL, and then let users link to an external reasoning engine. In other words, the semantic wiki community is still exploring the multiple points of junction between web 2.0 wiki aspects and semantic web frameworks capabilities. The next section surveys and compares contributions in this domain.

2. Semantic Wikis

Many semantic wikis are under development, and we focus here on those related to semantic web research. We do not consider others, such as FreeBase[4] (a commercial wiki) or OmegaWiki[5], which are less relevant to semantic web research than to discussions about user interfaces and structured data. While many of the wiki engines presented in this section are working prototypes, some, e.g., Semantic Media Wiki, have already been deployed in large scale applications. These, based on well-known existing wiki engines like JspWiki or MediaWiki, build on the stability, performance and robustness of these engines. In addition, the state of the art presented here can only be a snapshot of the contributions and versions available at the time of writing.

Looking at the state of the art, we can distinguish between approaches considering "the use of wikis for ontologies" and approaches considering "the use of ontologies for wikis" (few engines merge both approaches).

Most of the current projects on semantic wikis fall into the first category; *i.e.*, they consider wiki pages as concepts and typed links (in the page content) as object properties or data properties. In this model, called a "*wikitology*" [20], the wiki becomes the front-end of the ontology maintenance system.

One of the first wikis to fall into this category is Platypus [10] which imposes separately editing the metadata for each wiki page in a "Wiki Metadata page". It supports basic ontology editing but without consistency checking between the annotations and the ontology. It does not come with a reasoning engine and supports only basic queries. Semantic metadata are used for improving navigation but users have to switch between editing normal text and editing semantic annotations as these activities are done using two distinct text-based editors. Other wikis like SHAWN [6] offer similar features. Some wikis in this category address Platypus' shortcomings by allowing semantic annotations directly in the text of the page, usually as typed links.

Rise [20] also falls into the first category: the ontology used by the community is edited via the Wiki itself and a set of naming conventions is used to automatically determine the actual ontology from the Wiki content. A proprietary language is used for describing the metadata and RDF exportation is possible. Semantic information is used for navigation and consistency checks. The ontology is rebuilt every night to take into account added and modified wiki pages.

Rhizome [49] supports a modified version of WikiML (ZML) that uses special formatting conventions to make semantic properties directly explicit in the page content. Pages are saved in RDF and another editor can be used to edit the RDF directly. Rhizome authors admit that this feature is dangerous as one can break the wiki behavior by entering bad RDF. To mitigate the inherent dangers of this level of openness, Rhizome provides fine-grained authorization and validation alongside the use of contexts. It is not clear how metadata improve the wiki behavior. There is no advanced search and no help for navigating the wiki so far. RDF-Wiki [45] is similar to Rhizome in that it allows RDF annotations for external processing.

Semantic Media Wiki[6] [35][51] is based on MediaWiki. In contrast to Rise, typed links can also be used for specifying attributes of the page. For example, the following text: *San Diego is a [[is a::city]] located in the southwestern corner of [[is located in::California]]* establishes the facts "San Diego is a city" and "San Diego is located in California". While the text *Its coordinates are [[coordinates:=32°42'54"N, 117°09'45"W]]* defines an attribute named "coordinates". These data are used for faceted navigation. Semantic Media Wiki translates these metadata into RDF and support for the KAON2 reasoning engine exists as a proof of concept [52]. Other semantic extensions of MediaWiki are available [39] but are still at an early stage of development.

Makna [21] is based on JSPWiki and provides semantic extensions as typed links. It comes with the JENA reasoning engine that allows queries. Its text-based editor proposes extra HTML forms (with auto-completion) to query the semantic engine and to look for concepts/properties/relationships. This is useful in the case of a large ontology.

WikSar [5] enables users to enter semantic annotations from the wiki text editor using WikiWords. For example: if a page named "PrinceHamlet" contains a line "FigureBy: WilliamShakespeare", it can be seen as an RDF statement stating that the figure of Prince Hamlet was created by William Shakespeare. By listing the primitives used in all such embedded statements, an ontology can be extracted from the content of the Wiki. The editor is text-based and proposes no help of any kind to the user nor any consistency check. As pages are saved, the metadata are used to propose faceted navigation. WikSar supports queries in RDQL and SPARQL, and queries can be embedded in wiki pages or templates. A distinctive feature of WikSar is the graph visualization and navigation tool that can be used for exploring the wiki through its network of metadata.

[4] http://www.freebase.com (previously named MetaWeb and QuicksilverWiki).

[5] http://www.omegawiki.org

[6] We called it SeMediaWiki in Table 1.

	Platypus	Shawn	IkeWiki	Rise	Rhizome	SeMediaWiki	Makna	WikSar	AceWiki	SweetWiki	SWiM	OntoWiki	POWL
reuses an existing engine	no	no	no	no	no	MediaWiki	JspWiki	no	no	no	IkeWiki	MediaWiki	no
wiki object model	no	no	no	no	no	no	no	no	no	yes	yes	no	no
annotations in content	no	yes	no	yes	yes or edited separately	yes	yes	yes	yes	yes	no	*NA*	*NA*
separated editor for annotations	yes	no	For typed links	no	yes	no	no but external editor for statements	no	no	no	no	*NA*	*NA*
assisted annotations	yes	no	yes	no	no	no	yes	no	yes	yes	yes	no	no
social tagging	no	no	yes	no	no	no	no	no	no	yes	yes	no	no
edit ontology in wiki pages	no	yes	yes	yes	yes	yes	yes	yes	yes	no	yes	yes	yes
ontology edition features	basic RDF editing, text based, relations edited via the wiki medatada page	typed links	yes, some support	typed links	using ZML	typed links	typed links + statements	typed links	yes, in ACE	embedded ontology editor	yes, some support	yes	yes
representations languages	RDF/S & OWL	proprietary	OWL	proprietary, RDF export possible	RDF/S + ZML	RDFS	RDF/S, OWL	RDFS	ACE	RDFa, RDF/S, OWL lite	RDF/S, OWL	RDFS	RDFS/OWL
loading saving ontologies	yes	RDF export	yes	ontology is exported nightly	yes	export RDF	yes	export RDFS	no	yes	yes	export RDFS	RDFS/OWL export
queries	no	no	SPARQL	basic	no	WikiML extension	no	wikiML, RDQL, SPARQL	no	SPARQL	SPARQL	no	RDQL
reasoning engine	no	no	Jena	no	no	external (Kaon)	Jena	no	no	Corese	Jena	no	no
versioning for pages	yes	no	yes	no	yes	yes	yes	no	no	yes	yes	yes	yes
versioning for metadata	no	no	no	no	through contexts	within page	within page	no	no	within page	no	yes	yes
how metadata are exploited	navigate	navigation	navigate, render, search	for exporting an ontology	navigate, render, search	navigation, search	navigate	navigate, render	defining concepts	all wiki features render navigate,	navigate, render	to build an ontology	to build an ontology
WYSIWYG editor	no	no	yes	no	no	no	no	no	yes	yes	no	no	HTML only
Persistence	text	text	DB	text + DB	DB	DB	text + DB	text	text	XHTML + RDFa	DB	DB	DB
Discussion pages	no	no	yes	no	yes	yes	yes	no	no	no	yes	yes	no
Security/access rights	no	no	yes	yes	yes	yes	yes	no	no	yes	yes	yes	yes
Notification	no	no	rss	no	no	yes	yes	no	no	not yet	rss	yes	no
AJAX-based GUI	no	no	yes	no	no	no	yes	no	yes	yes	no	no	no
programming language	java	perl	java	java	python	PHP	java	perl	java	java	java	PHP	PHP
Licence	GPL	N/A	GPL	commercial	GPL	GPL	GPL	from author	N/A	LGPL like	GPL	GPL	GPL

Table 1: Summary of the semantic wikis's features

Typed links are powerful but one has to remember each concept, relation, and property before typing it and this is not practical. AceWiki[1] goes further. With AceWiki one can add and modify sentences written using the ACE language (Attempt to Controlled English [3]), through the use of an interactive editor. The editor is aware of the background ontology, and provides guidance to the user by proposing only valid completions. Moreover, the editor can be used to extend the ontology by creating new concepts, roles and individuals. Therefore, it is also, de facto, a simple ontology editor.

The second family of approaches focuses on *"the use of ontologies for wikis*". IkeWiki [46][47] supports WYSIWYG editing of page content and metadata, as well as page tagging. The editor comes with some dynamic features like auto-completion on metadata. It requires an existing ontology to be loaded. Some support for ontology editing is provided. It uses Jena as a triple store and SPARQL engine, and metadata are used for navigation, search and page rendering. Annotations can be visualized in a frame next to the wiki page. SWIM, a semantic wiki for mathematical knowledge management is also based on an older version of IkeWiki's engine, but uses OMDoc for describing the pages and their annotations. In addition to IkeWiki's features, a WYSIWYG editor dedicated for OMDoc editing is planned.

Our tool presented in this article, SweetWiki also falls into this second category. It does not implement the wikitology model yet (but we have foreseen such an evolution). Its most original feature is the use of a "wiki object model", an ontology of the wiki itself, that describes the data model of the system so that the wiki can be queried within its own documents. For example, requests like "show pages and videos that talk about this subject" are made possible.

It supports the concepts of social tagging and folksonomy[7] in addition to external ontologies. SweetWiki and WikSar share many features like usage-driven ontology building, queries embedded in the wiki pages, editing of metadata and page content in the same editor. SweetWiki adds a reasoning engine and an extensible WYSIWYG editor for both content and metadata, (like IkeWiki or Makna). The SweetWiki editor is AJAX-enhanced, and annotating pages leads to instant gratification as users type: (a) users can see an instant display of faceted links that the annotation will add to the page; (b) an auto-completion mechanism suggests existing concepts from the ontology, related categories and number of pages sharing that annotation as an incentive to reuse existing tags. Furthermore, SweetWiki comes with complete user-friendly ontology supervising and editing tools. However, SweetWiki is not dedicated to *collaborative* ontology management like OntoWiki [34] or POWL [4] but we have planned to add such capabilities to the engine.

Looking at the summary in Table 1, we can notice that most semantic wiki engines are open source, that only a few of them use rich "Web 2.0" AJAX-powered user interfaces, or embed a reasoning engine.

In the next sections we will present the architecture of SweetWiki and detail some of its key features.

[7] A folksonomy can be seen as a shared vocabulary that is both originated by, and familiar to, its primary users. In SweetWiki it is encoded as an ontology which is growing as users tag pages, pictures, videos, etc. It can be edited using the embedded editor.

3. SweetWiki

The idea of SweetWiki is to revisit the design rationale of wikis, taking into account the wealth of new standards available for the web eleven years later to address some of the shortcomings identified through experience. SweetWiki *relies on web standards* for the wiki page format (XHTML), for the macros included in pages (JSPX/XML tags), for the semantic annotations (RDFa, RDF), for the ontologies it manipulates (OWL Lite), etc. It *improves access to information* with faceted navigation, enhanced search tools and awareness capabilities, acquaintance networks identification, etc. It also provides a single WYSIWYG editor for both metadata and content editing, with assisted annotation tools (auto-completion, checkers for embedded queries or annotations). It comes with an embedded ontology editor and a reasoning engine. Another interesting point is that SweetWiki *allows metadata to be extracted and exploited by other applications*.

By semantically annotating the resources of the wiki and by reifying the *wiki object model itself*, SweetWiki provides reasoning and querying capabilities. All the models are defined in OWL schemata capturing concepts of the wikis (wiki word, wiki page, forward and backward link, author, etc.) and concepts manipulated by the users (*users' folksonomy, external ontologies*). These ontologies are exploited by an embedded semantic search engine (CORESE[14]) allowing us to support the lifecycle of the wiki, e.g., restructure pages, to propose new functionalities, e.g., semantic search, user-profile-based monitoring and notification, and to allow for extensions, e.g., support for new medias or integration of legacy software.

In SweetWiki we have paid special attention to preserving the essence of a wiki: simplicity and social dimension. Thus SweetWiki supports all the common wiki features such as easy page linking using WikiWords, versioning, etc., but also innovates by integrating a WYSIWYG editor extended to support social tagging functionalities, embedded SPARQL queries etc., masking the OWL-based annotation implementation. Users can freely enter tags and an auto-completion mechanism suggests existing ones by issuing queries to identify existing concepts with compatible labels. Thus tagging is both easy and motivating (real time display of the number of related pages) and concepts are collected in folksonomies. Wiki pages are stored directly in XHTML or in JSPX format, embedding semantic annotations (RDFa and GRDDL) ready to be reused by other software.

In the next section we detail the architecture of SweetWiki, in section 5 we will focus on the way we implemented the support for social tagging in SweetWiki, including enhancements to the WYSIWYG editor as well as an embedded web-based ontology editor that can be used for editing, creating and managing ontologies, including the users' folksonomy built upon the tags. In section 6 we show how the SweetWiki framework turns SweetWiki into an "application wiki", a wiki that enables users to write small applications within wiki pages. We conclude in section 7 with future enhancements and positioning.

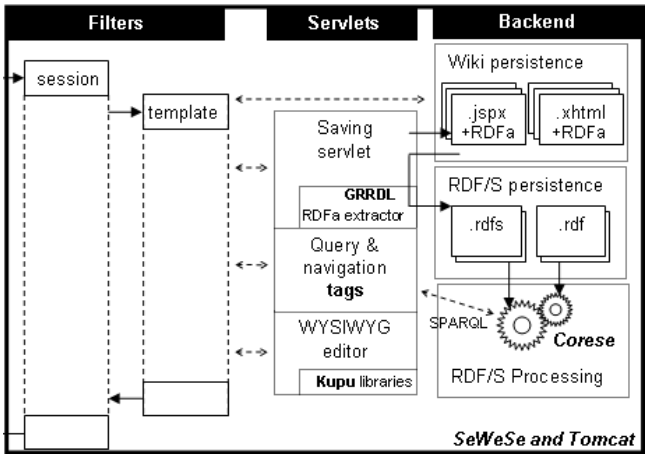

Fig 1. Architecture of SweetWiki in SEWESE.

4. SweetWiki Architecture

In this section we detail the different SweetWiki components: ontologies, semantic search engine, WYSIWYG editor and tagging support.

4.1 Ontologies in SweetWiki

Historically, wikis were Web sites where pages were organized around WikiWords, sometimes using other constructs such as WikiWebs[8] or Workspaces. To go beyond this informal hyperlink structure, we propose relying on semantic tagging and restructuring functionalities. To make explicit, manipulate and exploit such a structure we designed SweetWiki around a "*wiki ontology*" that describes the wiki concepts themselves. Furthermore, SweetWiki is also capable of taking advantage of external ontologies and folksonomies developed by its users through tagging.

4.1.1 The Wiki Object Model

SweetWiki relies on what we called "*The Wiki Object Model*", in other words: *an ontology of the wiki structure*. In many wiki engines, including several "semantic wikis", the wiki concepts are hidden in the *ad hoc* implementations. Following the ontology challenge [38] we made them explicit. In our case we defined in OWL Lite all the concepts, properties and relationships that we are using in the wiki itself. Concepts like "document", "page", "tag", "link", "backward link", "contributor", "version", "attached file", "attached picture", etc. are described in this ontology. The corresponding metadata are embedded in the pages themselves.
There are several advantages in using such a declarative Wiki Object Model. We can reason on it, e.g., use rules to complete, use inverse property to maintain the duality between forward and backward links, etc. We can extend and reorganize it, e.g., re-engineer the wiki structure, and we can build on it, e.g., interoperability between several wikis. We can generate widgets for helping the navigation, e.g., list related pages, SPARQL can be used to query all these metadata. The ontology of the wiki structure is maintained by the developers of the wiki.

4.1.2 The domain ontology (users' folksonomy)

SweetWiki supports social tagging. In the current version, pages and attached documents (pictures, videos, attached files) can be

[8] WikiWebs & Workspaces can be seen as folders of pages.

tagged from within the editor. The tags entered by users form a folksonomy. This will be detailed in section 5.

In order to ease navigation while maintaining the usual simplicity, we implemented the usual tag/keyword mechanism with a domain ontology shared by the whole wiki. By making this topic ontology explicit (we used RDFS) we can once again reason on it, e.g., find semantically close topics, or make complex queries, e.g., find pages according to the topics they were tagged with. We can modify it, e.g., tidy the ontology, merge equivalent concepts, declare hierarchical links, import domain ontologies, etc.

The domain ontology is enriched directly by the users and may be restructured by volunteers of the community to improve the navigation and querying capabilities. SweetWiki comes with an ontology editor that can be used by any user in order to organize the tags in the folksonomy (see more details in section 5.2).

4.1.3 Support for External Resources

Other ontologies may be added at runtime by privileged users and become immediately accessible to users for SPARQL[9] queries, tagging, and integration. If the wiki is used in a domain, e.g., biology, for which some ontologies are already available in RDFS or OWL Lite, these ontologies can be loaded into the underlying semantic web server of SweetWiki. This is also an excellent way to bootstrap the pool of tags and it facilitates interoperability by fostering the reuse of existing ontologies. In addition we foresee that a query could be embedded in a page but directed to other SPARQL servers, thus allowing users to include results from external sites running SweetWiki instances.

4.2 Semantic Search Engine and Web Server

As shown in Fig. 1, the implementation of SweetWiki relies on the CORESE[14][13] semantic search engine for querying and reasoning and on SEWESE [27], its associated web server extension that provides high level APIs and JSP tags to implement ontology-based interfaces as well as a set of generic functionalities (security management, ontology editors, web application life cycle, etc.).

There are many other papers and online resources[10] describing the underlying CORESE engine; we do not go into such details here. However, a concise description could be: a semantic search engine implementing the whole SPARQL syntax (including the regex filter function) with some minor semantic modifications (OPTIONAL statements are post processed, nesting OPTIONAL and UNION has some restrictions), also implementing RDF and RDFS entailments, datatypes, transitivity, symmetry and inverse property entailments from OWL Lite. CORESE also provides a production rule engine with a synthetic syntax close to SPARQL that is used in particular to implement rule-checking OWL restrictions against the base. CORESE does not include a tableau algorithm, i.e., it does not address ontology classification and considers that this was already done beforehand. CORESE also offers several extensions, e.g., GROUP BY and COUNT operators, for details see [13] [14] [15] [16][17][18] [22] [23] [24] [25] [26] [30] [31].

The server relies on a standard web application architecture: filters manage the session, e.g., authorization, user profiles, etc., and the template of the site (headers, trailers). Pages are directly available in XHTML or JSPX for browsing. A servlet handles saved pages and a set of JSP tags provide high level functionalities (e.g., submit a SPARQL query and format the result with an XSLT style sheet). JavaScript libraries are served to provide a WYSIWYG editor.

4.3 The Editor

SweetWiki uses an XHTML WYSIWYG editor based on Kupu. This editor is used both for content and metadata editing. The original Kupu editor has been considerably extended in order to support semantic extensions and metadata editing:

- SPARQL queries can be embedded into a page being edited and they are translated at save time into JSP tags from the SEWESE tag library. From the editor, SPARQL queries can be tested and validated before being inserted into the page, as illustrated by Figures 2 and 3.
- Linking to other internal pages is easy using AJAX-powered wizards that execute SPARQL queries for getting the list of existing documents.
- Tagging a page or parts of a page such as included pictures, videos or attached files also relies on AJAX calls from within the editor e.g. when entering a tag, a SPARQL query is issued and an auto completion mechanism proposes existing tags.

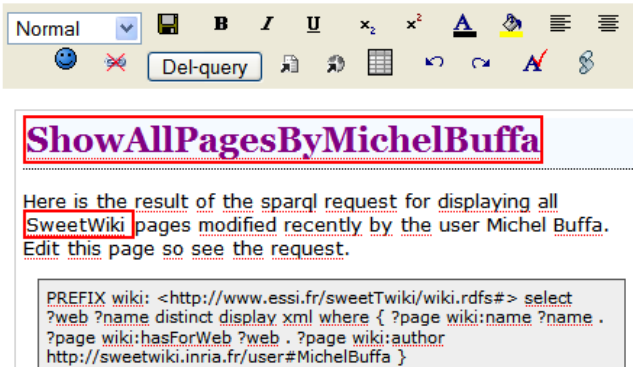

Fig 2. Example of an embedded SPARQL.

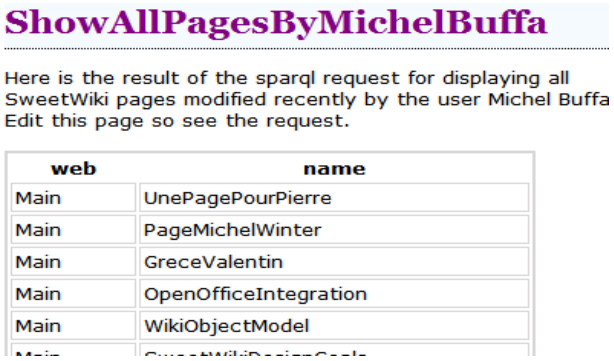

Fig 3. Page showing the result of the query[11]

Some metadata are included by the editor dynamically in the DOM of the page and others are added at save time (e.g. date).

[9] http://www.w3.org/TR/rdf-sparql-query/

[10] http://www-sop.inria.fr/edelweiss/wiki/wakka.php?wiki=Corese

[11] The current implementation uses WikiWords for page names and internal page linking but a wizard is provided by the WYSIWYG editor for standard hyperlinks to pages.

The RDF model has an XML syntax but it is currently impossible to validate documents that contain arbitrary RDF/XML tags and therefore it is a problem to import RDF/XML into other markup languages such as XHTML. On the other hand, the external annotation of a document in RDF/XML can result in significant data duplication between the actual annotated resource and the RDF/XML annotation. For the sake of maintenance, conciseness, and encapsulation it is often better to add RDF to a document without repeating the document's existing data. We use the RDFa[12] syntax for embedding these metadata into the XHTML of the page. RDFa proposes a solution to augment existing markup with metadata, using class and property types defined in RDF Schemas, combined with the existing content from the host language.

Figure 4 shows the RDFa code corresponding to a picture embedded in a SweetWiki page, tagged with the keyword "holidays". The WIKI namespace corresponds to the wiki ontology described in section 4.1.1.

```
<span about="/wiki/pub/102_9718.jpg">
    <img class="image-inline" alt="102_9718.jpg"
src="/wiki/pub/102_9718.jpg" />
    <link href="WIKI:Image" rel="rdf:type">
    <link href="http://www.inria.fr/acacia/users-
ontology#holidays" rel="WIKI:hasForKeyWord">
</span>
```

Fig 4. Example of RDFa code embedded in a SweetWiki page

Contrary to external RDF annotations, this approach is inline with the wiki spirit where everything is done within the page itself: anything can be copied and pasted in one place (the wiki page) even using a WYSIWYG editor. With RDFa we have both page data and metadata served in the same standalone file (XHTML) and pages can be crawled by external applications or saved by users using their browser without any loss of information.

GRDDL is a mechanism for getting RDF data out of XML and XHTML documents using explicitly associated transformation algorithms, typically represented in XSLT [12]. We use it to extract metadata from the wiki pages and this scenario inspired one of the motivating use cases proposed for GRDDL[13].

Therefore, when saving a page several operations occur:

1. The metadata are extracted from the page and translated into RDF/XML using a GRDDL transformation (XSLT stylesheet)
2. The RDF is loaded in the CORESE semantic search engine,
3. Another stylesheet is applied for transforming the dynamic content of the page into JSP tags (e.g. SPARQL queries).

Pages are served as is by the Tomcat server; no extra processing is needed as the pages are standard JSP or XHTML pages. For example, if a page contains an embedded request as in the previous example, the JSP tag will be executed, the request will be sent to the semantic search engine and the results will be inserted dynamically as an XHTML table, as shown in figure 3.

The editor can only edit pure XHTML whereas some wiki pages are in JSPX format. In fact only the "dynamic parts" of the pages, like embedded SPARQL queries are encoded using some non XHTML tags, the rest of the page is standard XHTML. In order to make the page editable using the editor, we apply a style sheet that turns these tags into their "XHTML view" (typically using some … XHTML tags). It is exactly the reverse operation to that performed in step 3, when the page has been saved. A typical example is a SPARQL query that has been entered in the WYSIWYG editor. It is coded in the DOM using some XHTML attributes which are then turned into JSP tags provided by the SEWESE toolkit i.e. the following request:

```
<span property="sparql:query"
xslt="/xsl/query_result_as_table.xsl">PREFIX wiki:
&lt;http://www.essi.fr/sweetTwiki/wiki.rdfs#&gt;
select ?web ?name distinct display xml where {
?page wiki:name ?name . ?page wiki:hasForWeb ?web
. ?page wiki:author
http://sweetwiki.inria.fr/user#MichelBuffa
}</span>
```

... is translated into a JSPX tag from the SEWESE tag lib:

```
<sew:query query="PREFIX wiki:
&lt;http://www.essi.fr/sweetTwiki/wiki.rdfs#&gt;
 select ?web ?name&#10;distinct display xml
 where {
 ?page wiki:name ?name .
 ?page wiki:hasForWeb ?web .
 ?page wiki:author
http://sweetwiki.inria.fr/user#MichelBuffa }"
xslt="/xsl/query_result_as_table.xsl"/>
```

Such a declarative approach makes extension extremely easy. Imagine wanting to add support to embed videos - the model is easy to extend. Here are the steps that need to be performed:

1. Define in the Wiki Object Model (the wiki ontology) the concept of "video file", link it to the concept of page, and add any needed property (video length, codec, filename, etc.)
2. Add a plugin to the editor (button + callback for inserting the XHTML code in the page (using some) + inserting the corresponding metadata in RDFa).
3. Update the two style sheets for converting back and forth from the XHTML view to the dynamic JSP tag view.
4. Update the tag lib so that the corresponding JSP tag will generate the XHTML code for embedding a viewer for the video file format.

Now in any page of the wiki you can embed SPARQL queries for querying all the videos in the pages created by a given user.

4.4 Office Document Integration

SweetWiki comes with a module to import Open Office or Microsoft Office documents (in the current version we support word processor files as well as spreadsheet files). The documents are automatically translated into SweetWiki pages (including the pictures, etc.) that can in turn be edited, tagged and shared. In the future, metadata available in these documents could be exploited as well. An Open Office server and ad hoc conversion modules are used behind the scenes for converting the office documents.

5. Semantic Web Frameworks and Tagging

In this section we decribe how tagging was implemented in SweetWiki using semantic web frameworks.

5.1 Tagging a Wiki

SweetWiki supports social tagging. Users can tag pages, pictures, attached files, etc. As in [7] and [41], we propose a mixed approach in order to "organize the tags". We link the tags together within a folksonomy described using the semantic web languages, where tags are organized in a hierarchy and related to one another using relationships like `subClassOf`, `seeAlso`, etc. Gruber goes further and has proposed in [32] to define "an Internet

[12] http://www.w3.org/TR/xhtml-rdfa-primer/
[13] http://www.w3.org/TR/grddl-scenarios/

ecology" for folksonomies *i.e.*, an ontology for describing folksonomies. Similarly, we believe that social tagging minimizes cost and maximizes user participation, so we support social tagging in SweetWiki, but we also think that these tags must be organized. The system we have implemented helps users build a useful folksonomy while relying on standard semantic web technologies for organizing and maintaining the folksonomy. SweetWiki uses folksonomies and social tagging as a better way to categorize the wiki documents [33] [43]. Other efforts have been conducted recently for creating a standard ontology of tags, such as the RFC proposed by Newman [40], or by including a definition of tags within a larger ontology as in the SIOC framework[14]. In our case, we define the tags in the Wiki Object Model (see section 4.1.1).

As described in Section 4.6, SweetWiki integrates a standard WYSIWYG editor that we extended to directly support semantic annotations following the "social tagging" approach. Users can assign concepts to pages and use an editor to modify the concept hierarchy.

As shown in Figure 6, when editing a page the user can *freely* enter some keywords in an AJAX-powered text field. As the user types, an auto-completion mechanism proposes existing keywords by issuing SPARQL queries to the semantic web server in order to identify existing concepts with compatible labels and shows the number of other pages sharing these concepts as an incentive to use them.

Furthermore, parent categories are also displayed in order to address the ambiguity of homonymy. With this approach, tagging remains easy (keyword-like) and becomes both motivating and unambiguous. Unknown keywords are collected and attached to new concepts to enrich the folksonomy. Later on, community experts may reposition them in the ontology, edit them, etc. The feedback coming from the tags is useful for improving the ontology and for discovering a community's vocabulary.

The CORESE engine is used to generate faceted navigation widgets. The semantics of the tags is used to suggest related topics, query the engine on similar pages using SPARQL queries, notify interested users, etc. (see Figure 7). When a SweetWiki document is requested by a web browser, templates are used to integrate the faceted navigation widgets around the page content. These templates may be changed like the skins in TWiki[15].

5.2 Embedding a web-based ontology editor

This editor can be used by any user for maintaining and re-engineering the folksonomy and for importing, creating and updating external ontologies. By letting users themselves organize and update the folksonomy, it has been shown that search and navigation are generally improved [33] [43]. Even if a small percentage of users contribute to the folksonomy's maintenance, the whole set of users will benefit from every improvement. Experiments conducted at *Electricité de France*[42], an energy company, have confirmed the promise of such an approach (even if in that case only administrators could update the folksonomy's structure).

Supervising tools are integrated into SweetWiki by relying on the semantic web server SEWESE. They are used to monitor the wiki activity itself running SPARQL queries over the metadata e.g. usage frequency for tags (Figure 5), new tags, orphan pages, etc.

In order to maintain and re-engineer the folksonomy, SweetWiki also reuses web-based editors available in SEWESE. In our examples we tagged some Java courses, using a Java ontology. Selecting this ontology in the editor, one can add, remove and edit concepts (labels, parents, comments) whose labels are used to tag the pages. In particular, if a concept has been recently added it may be repositioned in the hierarchy by the user (Fig 8).

Using these editors, the folksonomy and the annotations may be updated. For instance, community experts can pick a pair of tags and declare semantic relations between them such as `subClassOf`. They may also merge concepts when two tags are synonymous, etc. Enhancements of the ontology seamlessly improve content sharing: search and faceted navigation benefit directly from the updates (e.g., new suggestions appear in the navigation widgets). The way the system is designed, versioning cannot break the annotations. If a tag is suddenly missing from the ontology but still used in pages it is just tacked on as a new tag and if many pages exist with the old tag (pages are not touched in the tag editing process), the tag reappears (with a high number of tagged pages, encouraging other people to use it). Re-engineering the ontology is a way of refactoring the wiki - new links and groupings appear as the ontology is enriched.

From the end-users' interface, only basic manipulations are allowed on the ontology, essentially corresponding to RDFS expressivity plus algebraic characteristics of properties and inverse of property declaration. The hierarchy of properties can also be edited, however new properties require new widgets or plug-ins to generate the corresponding triples. For instance, when the property "interested by" was added, we developed a widget for the homepages of the users to generate the corresponding annotations.

Keyword	Subclass Of	Pages
sweetwiki	wiki_enginewikisoftware	12
palette	.	10
model	.	8
CoP	communitypalette	7
Competency	palette	6
ux11	CoPcommunitypalette	5
mbds	filiere_de_formation	5
ejb	NewConcept	4
Collaboration	palette	4

Fig 5. Tags sorted by popularity (results computed by a SPARQL query using the COUNT extension provided by CORESE)

[14] http://sioc-project.org/
[15] http://www.twiki.org

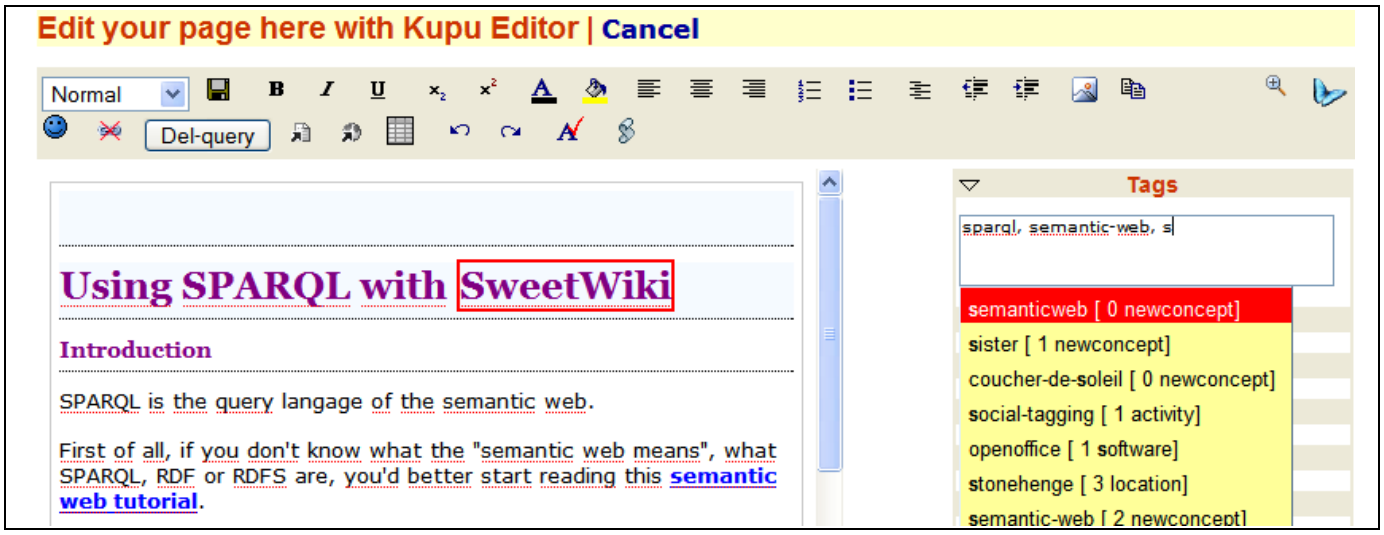

Fig 6. Tags are suggested as the user enters keywords. The number of pages using each tag and the related category are displayed.

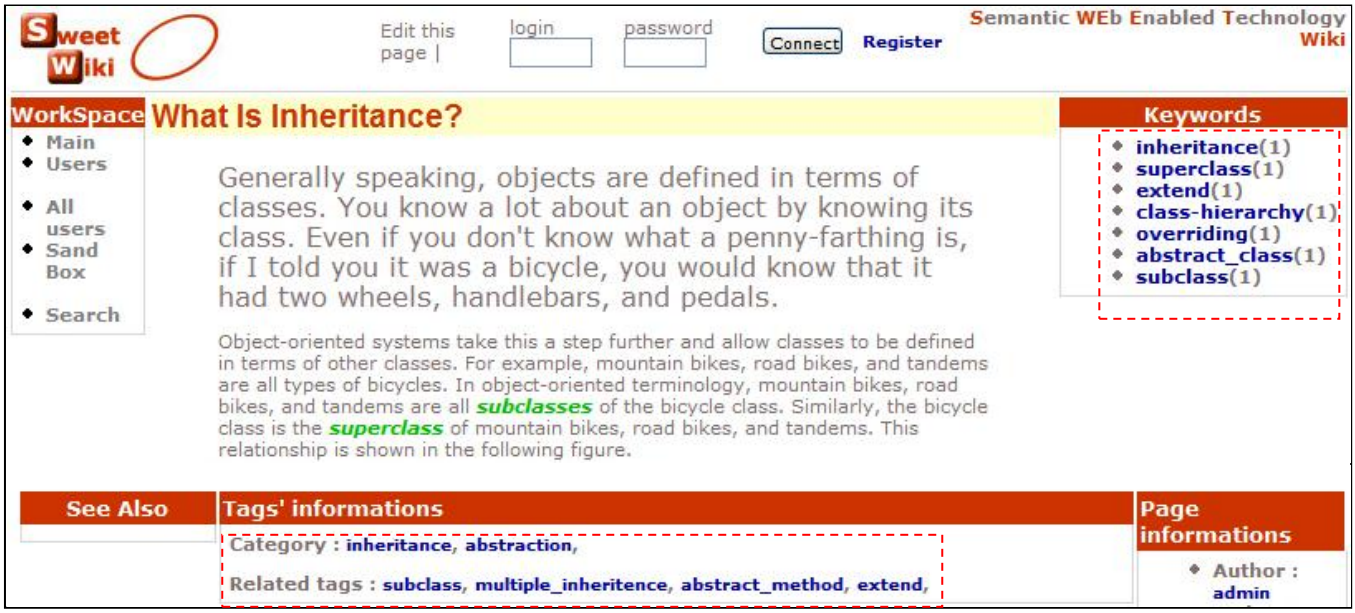

Fig 7. Faceted navigation links extracted from the tags

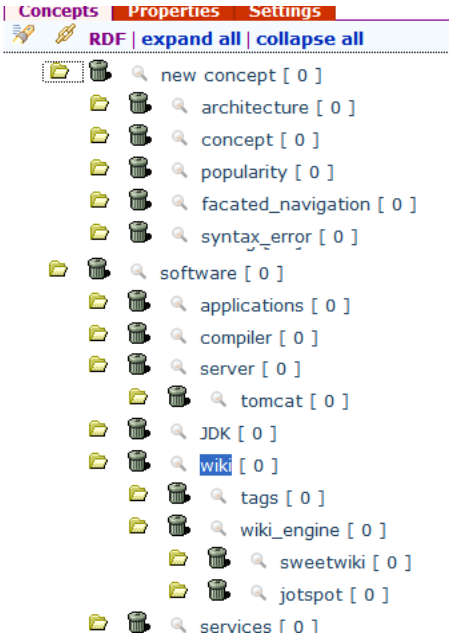

Fig 8. The ontology editor, with the users' folksonomy..

6. SweetWiki is an Application Wiki

Several wiki engines like TWiki, JotSpot, Confluence or XWiki are called "application wikis" in the sense that the WikiML language used for formatting the documents includes some very powerful macros. These macros make the writing of simple table-based applications easy, with no need to set up a database or for writing more than a few lines of code. None of these wikis use a rich editor when writing wiki applications. Only a few users have the skills to write such wiki applications (see [11] and [8] for a survey) but this feature makes the wiki a powerful tool for adding "small applications" without having any coding knowledge.

In SweetWiki we can embed dynamic content like queries, in the pages, directly from the WYSIWYG editor and query any part of the wiki (through the Wiki Object Model). We showed in previous sections that by associating XHTML … tags and JSP tags we can easily make any dynamic content/macro editable in the WYSIWYG editor.

By allowing dynamic editing and querying, any user can enter complex queries in any page and provide some "application pages". The screenshot in Figure 9 shows a search page provided by the SweetWiki standard distribution - the page is editable and users can modify the SPARQL queries to suit their needs. The example shows the results of a search on the tag "sweetwiki": pages and pictures tagged with this tag are shown but also related tags are extracted from the user's folksonomy (more on that in section 5). An important point is that such a page has been written as a "regular" SweetWiki page, using the WYSIWYG editor. Users can upload XSLT style sheets in order to customize the presentation of the results of the queries. In the current version the WYSIWYG editor does not provide assistance for building the query except for a SPARQL syntax validation and a result preview. Another limitation is the lack of a rich form editor to parameterize the queries.

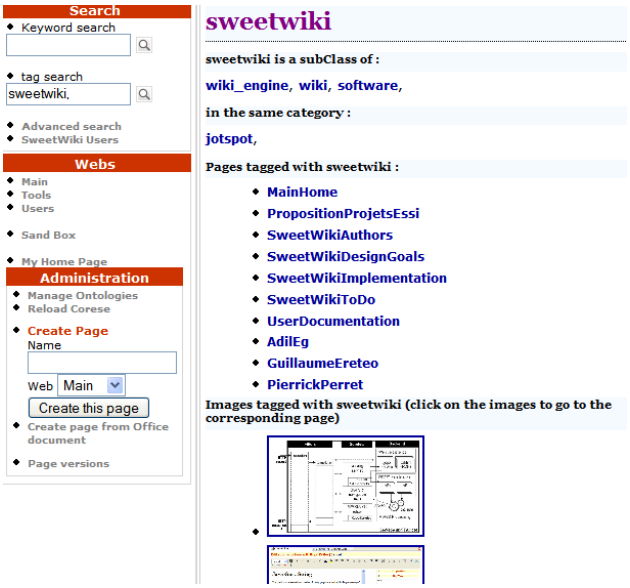

Figure 9: Search pages can be edited in the WYSIWYG editor.

When a user registers, a home page is automatically created. This home page's metadata are a bit different from the ones embedded in other standard pages as they represent the profile of the user. In the current version a user can enter a list of tags corresponding to the topic he/she is interested in. For example, a user who says he/she is interested in "wikis" will have some suggested links proposed or mailed to him. As in any query resolution submitted to CORESE, inheritance is taken into account. If some pages or pictures are tagged with subclasses of the "wiki" tag, we assume that these pages "talk about wikis" and they will be suggested as well. Figure 10 shows the home page of user in editing mode. Figure 11 shows a part of the home page that is rendered once saved. In this approach users can anticipate new topics and define interest a priori even before a page was created on a given topic. In classical wikis, you may be notified of a page being modified; here you can also be notified when a page is created, when someone uses a specialization of a tag you are interested in, etc.

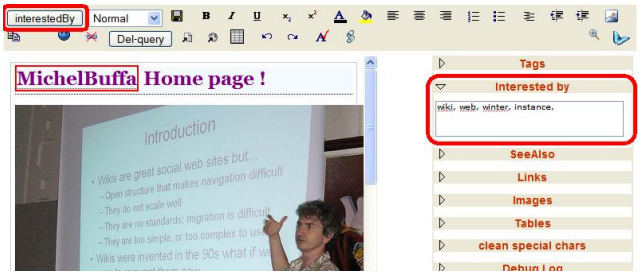

Figure 10: A homepage, tagged with personal interests.

Last updated pages containing the tags which interest you

WikiPage	Web	Author	Description	Last update	Keyword
PropositionProjetsEssi	Main	GuillaumeEreteo	Description - not implemented	2006-11-15	sweetwiki
SweetWikiToDo	Main	GuillaumeEreteo	Description - not implemented	2006-11-15	sweetwiki
MyleneLeitzelman	Users	MyleneLeitzelman	Description - not implemented	2006-11-13	wiki
MainHome	Main	AdilElGhali	Description -	2006-11-10	sweetwiki

Fig 11: Suggested pages related to a user's topics of interest.

It is also interesting to analyze users' behavior: what they do, what they know, what they want. For example, by keeping track of pages created and modified, of documents tagged and watched, and more generally by analyzing users' behavior over time, it is possible to build acquaintance networks or communities of interest. We have thus updated the wiki object model to store all the necessary metadata. Then by reasoning on these metadata with simple rules like "if someone watches a page tagged with a topic (e.g., "ajax") then they may have some knowledge about the parent topic in the folksonomy, (e.g., "web 2.0"), we can, for example, give a sorted list of persons who have some knowledge about a topic. So far, we have implemented new navigation and search functionalities based on these ideas:

- Look for the most active person w.r.t a given topic to help us find experts.
- Build behavior similarities between users to foster communities of interest.
- Infer relationships between tags that are not explicit in the folksonomy, such as co-occurrence or context sharing.

7. Discussion

By nature, a Web application requires taking into account its semiotic dimension (as a meaningful system mobilizing signs of all types to build representations for humans), its pragmatic dimension (as a semiotic system with multiple usages which influence its interpretation) and its social dimension (as a virtual space of interaction). In our opinion, nothing in the semantic Web initiative is opposed to taking these dimensions into account quite to the contrary we might add. The official semantic web activity page says "The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries"[16]. The core motivating scenario of the semantic web initiative thus is assistance to collaboration.

Over the past eight years, semantic web researchers have proposed pivot languages to represent and exchange data, but they have prescribed or restricted neither the use of these languages nor the methods to generate the data they convey. Opposing ontology and folksonomy is like opposing a "cake" and a "baked cake" - they are not at the same level of abstraction. Folksonomies are defined by the way they are obtained (social tagging). Ontologies are defined by their content (a representation of a conceptualization), and not by the way they are obtained. Furthermore, nothing in the objectives or formalisms of the semantic web is opposed to taking into account the social dimension of the web.

Another important point that is often missed is the notion of domain of formalization. In a semantic web application, the domain to be formalized in RDF/S or OWL is not always the application domain since formalization is primarily specified by the task. Thus if the task is, for instance, assistance with the alignment of several medical terminologies, perhaps formalization will focus on linguistic primitives (e.g. term, synonymous with, hyponym of, acronym of, etc.), allowing the representation and comparison of the various terminologies. In other words, we do not inevitably find the notions of our domain of application in our RDFS schema. Building semantic web applications in the automotive industry doesn't necessarily mean we will find the concept of "car", "windshield" or "engine" in our ontology. In a Web 2.0 application this means that the use of semantic web frameworks does not imply that the domain of application will have to be formalized. As an example, if I want to allow social tagging for a music bank I don't necessarily have to formalize music categories in RDFS but I can choose to formalize the domain of social tagging and declare the notions of "tag", "tag cloud", etc.

Another dangerous misconception is to consider that ontology-based solutions are necessarily centralized and/or monolithic solutions with frozen schemas. Multiple schemas, namespaces and equivalence relations are examples of core mechanisms of semantic web formalisms that clearly show that the semantic web vision is that of a distributed, decentralized, integration system. Moreover, to represent and publish an ontology in RDFS or OWL does not imply that it is now set in stone. The semantic web perfectly acknowledges the existence of a life-cycle of ontologies (e.g., relations of equivalences between two ontologies in OWL, best practices to choose URIs to manage the evolution of a concept, etc.).

Finally, designing an application using semantic Web frameworks does not imply building a solution only with the tools of the semantic web. On the contrary, a proprietary application which manages an electronic calendar is an application of the semantic web if it simply makes it possible to export and to import its data in one of the languages of the semantic web. It doesn't need to have a rule engine, a Prolog virtual machine, some tableau algorithms or a projection operator implementing SPARQL. The only effort which is required from it is to do the only thing towards the interoperability which cannot be done by something else to make explicit its data structures and the conceptualization on which it is based. It is the old challenge of ontologies but with results at the scale of the Web.

To summarize the overall scenario explored in SweetWiki, we have proposed an innovative approach that allows users to edit wiki pages and tag them using a shared and explicit conceptualization, but one that is hidden behind the scenes. In addition community experts can check the underlying model being built, look at the tags proposed by the users and (re)organize them. If this happens, annotations that users have entered are not changed, but faceted navigation and search based on semantic queries are improved by new links.

SweetWiki is available online[17] and is currently being used in several projects to share information among participants. These communities use it as the collaborative platform mainly for coordinating researchers' work. Indeed, our system is oriented at both non technical users and experts:

- Palette[18]: a European project on tools and methods to assist the learning in communities of practice. We started monitoring the behavior of communities of practice in Palette (section 3 from [28]) and their feedback and requirements (section 4 from [28]). Early usability results

[16] http://www.w3.org/2001/sw/

[17] http://argentera.inria.fr:8080/wiki

[18] http://palette.ercim.org/

for SweetWiki indicate particular interest in tagging without editing pages, semi-automatic means to organize the tags, ergonomic enhancements in interfaces, and advanced notification mechanisms.

- E-Wok[19]: a national research project on CO_2 capturing techniques requiring collaboration between geology and petroleum institutes.
- Informal tests: we have carried out experiments with children, teachers, students and secretaries asking them to use it for one of their daily tasks.

We are currently monitoring several experiments:

- in one of them we were asked to add options to lock parts of the folksonomy when the classification is judged satisfactory (teachers contributing to the wiki worried that the folksonomy may become degraded over time);
- in another case (a robotics company using SweetWiki for its intranet as well as its public community site) we decided not to advertise the presence of a folksonomy editor. After a few months we looked at the vocabulary that had emerged and together with a leading user of their community we started to reorganize it. We are observing the reaction of the users vis-a-vis the reorganization induced in the wiki, e.g., new suggested links. This experiment is still at an early stage.

In these cases, the folksonomies built are "light ontologies" or intermediate representations that could be used as starting points to generate more formal ontologies.

With SweetWiki, occasional users as well as experts can use the same tool for writing small wiki applications or for improving the ontology being used with the embedded ontology editor. Support for external ontologies and the possibility of embedding dynamic content such as SPARQL queries in the pages, turns SweetWiki into a "semantic web application platform". Even wiki core concepts like pages, links, pictures, authors, etc. can be extended and queried, relying on the *wiki object model* in OWL Lite.

A number of extensions are currently under consideration:

- *Develop more applications within the wiki (as presented in section 6), improve the different editors,*
- *Natural language processing for automatic tagging*: several wikis are starting to analyze the text of wiki pages to suggest potential keywords. Seamless suggestion of metadata could be achieved by applying natural language processing techniques to (semi-)automatically derive keywords from the existing content and its context; even light approaches like shallow parsing can be of great interest to improve usability.
- *Complete versioning*: supporting not only the versioning of textual content and semantic annotations, but also of underlying ontologies, and synchronizing all of these versions;
- *Collaborative folksonomy management*: providing groupware to assist the distributed lifecycle of ontologies. Here the *wikitology* approach seems only natural and we need tools to implement it both efficiently and in a user-friendly way.

This last point brings us back to the two-way vision *wikis for ontologies and ontologies for wikis*. This division of the current approaches is only symptomatic of the early times of semantic wikis. In the long-term future semantic wikis will merge these two approaches as two facets of the same coin as some projects have already started to do, the objective being to turn this two-way vision into a virtuous circle where users maintain the wiki and its ontology *at the same time* without any artificial distinction between them just as a side-effect of its use. We designed SweetWiki as an experimentation platform evolving incrementally towards this vision. Following the release of a first version of SweetWiki, we are working with several users' group to evaluate, test and improve our design rationale.

[19] http://www-sop.inria.fr/acacia/project/ewok/

8. REFERENCES

[1] AceWiki : http://gopubmed.biotec.tu-dresden.de/AceWiki/

[2] Adida B., Birbeck M., "RDFa Primer 1.0 Embedding Structured Data in Web Pages", W3C Editrors' Draft 2007/09/18, Embedding RDF in XHTML http://www.w3.org/2001/sw/BestPractices/HTML/2006-01-24-rdfa-primer

[3] "Attempt to Controlled English (ACE) www.ifi.unizh.ch/attempto/

[4] Auer, S., "Powl – A Web Based Platform for Collaborative Semantic Web Development", Proc. of the 1st Workshop on Scripting for the Semantic Web (SFSW'05), Hersonissos, Greece, 2005.

[5] Aumueller D. and Auer S. "Towards a Semantic Wiki Experience – Desktop Integration and Interactivity in WikSAR", Proc. of the Workshop on Semantic Desktop, Galway, Ireland, 2005.

[6] Aumueller D. "SHAWN: Structure Helps a Wiki Navigate" Proc. of the BTW-Workshop WebDB Meets IR, Karlsruhe, Germany, 2005. http://dbs.uni-leipzig.de/~david/2005/aumueller05shawn.pdf

[7] Bird. F. "Some Ideas to Improve Tags Use in Social Software, Flat Hierarchy versus Categories in Social Software", 2005.

[8] Buffa, M. "Intranet Wikis". Proc. of the Intraweb Workshop, 15th International Conference on World Wide Web, Edinburgh, Scotland, 2006..

[9] Buffa M., Gandon F., "SweetWiki : Semantic Web Enabled Technologies in Wiki", Proc. of the ACM Conference Wikisym, Odense, Denmark, 2006.

[10] Campanini S.E., Castagna P. and Tazzoli R. "Platypus Wiki: a Semantic Wiki Wiki Web", Proc. of the 1st Italian Semantic Web Workshop, Ancona, Italy, 2004.

[11] Chat C. and Nahaboo, C. "Let's Build an Intranet at ILOG Like the Internet! ", Proc. of the Intraweb Workshop, 15th International Conference on World Wide Web, Edinburgh, Scotland, 2006.

[12] Connolly D., "Gleaning Resource Descriptions from Dialects of Languages (GRDDL)", W3C Recommendation 11 September 2007, http://www.w3.org/TR/grddl/

[13] Corby O., Dieng-Kuntz , Faron-Zucker C., Gandon F. , "Searching the Semantic Web: Approximate Query Processing based on Ontologies", IEEE Intelligent Systems Journal, 21(1), 2006.

[14] Corby O., Dieng-Kuntz R, Faron-Zucker, C., "Querying the Semantic Web with the CORESE Search Engine", Proc. of the 16th European Conference on Artificial Intelligence (ECAI'2004), Valencia, Spain, 2004, IOS Press

[15] Corby O., Dieng R., Hébert C., "A Conceptual Graph Model for W3C Resource Description Framework", Proc. of the 8th International Conference on Conceptual Structures, Darmstadt, Germany, 2000, LNCS 1867, Springer-Verlag.

[16] Corby, Faron 2002 - O. Corby, C. Faron. "Corese: a Corporate Semantic Web Engine", Proc. of the International Workshop on Real World RDF and Semantic Web Applications, 11th International World Wide Web Conference, Hawai, USA, 2002.

[17] Corby O., Faron-Zucker C, Implementation of SPARQL Query Language based on Graph Homomorphism, Proc. of the 15th

International Conference on Conceptual Structures (ICCS'2007), Sheffield, UK, LNCS 4604, Springer Verlag.

[18] Corby O., Faron-Zucker C, "RDF/SPARQL Design Pattern for Contextual Metadata", Proc. of IEEE/WIC/ACM International Conference on Web Intelligence, Silicon Valley, USA, 2007.

[19] CREOLE A common wiki markup, http://www.wikicreole.org

[20] Decker B., Ras E., Rech J., Klein B. and Hoecht C. Self-organized "Reuse of Software Engineering Knowledge Supported by Semantic Wikis", Proc. of the Workshop on Semantic Web Enabled Software Engineering (SWESE), ISWC, Galway, Ireland, 2005

[21] Dello K., Tolksdorf R. and Paslaru E. Makna. Free University of Berlin. http://www.apps.ag-nbi.de/makna/wiki/About, 2005.

[22] Delteil A., Faron-Zucker C., Dieng-Kuntz R., "Le modèle des Graphes Conceptuels pour le Web Sémantique", RSTI L'Objet, 3(9), 2003.

[23] Delteil A., Faron-Zucker C., "A Graph-Based Knowledge Representation Language". Proc. of the 15th European Conference on Artificial Intelligence (ECAI 2002), Amsterdam, The Netherland, 2002, IOS Press.

[24] Delteil A., Faron-Zucker C., Dieng-Kuntz R., "Building Concept Lattices from RDF Graphs Annotating Web Resources", Proc. of the 10th International Conference in Conceptual Structures (ICCS2002), Borovetz, Bulgarie, 2002, LNCS 2393, Springer-Verlag.

[25] Delteil A., Faron-Zucker C., Dieng-Kuntz R., "Extending RDF(S) with Contextual and Definitional Knowledge", Proc. of the IJCAI Workshop "E-Business and the Intelligent Web", Seattle, USA, 2001.

[26] Dieng R., Corby O., "Conceptual Graphs for Semantic Web Applications", In: Proc. of the 13th Int. Conference on Conceptual Structures (ICCS'2005), Kassel, Germany, 2005, LNAI 3596, Springer-Verlag.

[27] Durville P., Gandon F., "SeWeSe : Semantic Web Server", WWW2007 Developers track, http://www-sop.inria.fr/acacia/personnel/Fabien.Gandon/research/www_2007_dev_track/

[28] El Ghali A., Tifous A., Buffa M., Giboin A., Dieng-Kuntz R., "Using a Semantic Wiki in Communities of Practice", Proc. of TEL-CoPs'07 Workshop, EC-TEL'07, Crete, Greece, 2007.

[29] Gandon F., "GRDDL Use Cases: Scenarios of extracting RDF data from XML documents", W3C Working Group Note 6 April 2007 http://www.w3.org/TR/2007/NOTE-grddl-scenarios-20070406/

[30] Gandon F., Corby O., Giboin A., Gronnier N., Guigard C., "Graph-based inferences in a Semantic Web Server for the Cartography of Competencies in a Telecom Valley", Proc. of ISWC, Galway, Ireland, 2005, LNCS 3729, Springer Verlag.

[31] Gandon F., Scientific Philosopher Doctorate Thesis In Informatics: "Distributed Artificial Intelligence and Knowledge Management: ontologies and multi-agent systems for a corporate semantic web". Defended on Nov. 7th 2002, INRIA and University of Nice - Sophia Antipolis. Doctoral School of Sciences and Technologies of Information and Communication (S.T.I.C.)

[32] Gruber T. "Ontology of Folksonomy: A Mash-up of Apples and Oranges". First on-Line Conference on Metadata and Semantics Research (MTSR'05). http://mtsr.sigsemis.org/, (2005).

[33] Hammond T, Hannay T, Lund B., Scott J. "Social Bookmarking Tools, a General Review", D-Lib Magazine, April 2005, 11(4),

[34] Hepp M., Bachlechner D. and Siorpaes K. "OntoWiki: Community-driven Ontology Engineering and Ontology Usage based on Wikis", Proc. of the International Symposium on Wikis (WikiSym 2005), San Diego, USA, 2005.

[35] Krötsch M., Vrandečić D., Völkel M. "Wikipedia and the Semantic Web" - The Missing Links, WikiMania 2005.

[36] Lange C., Kohlhase M, "Towards Scientific Collaboration in a Semantic Wiki", Proc. of the ESWC Semantic Wiki Workshop, Innsbruck, Austria, 2007.

[37] Majchrzac A., Wagner C., Yates D., "Corporate Wiki Users: Results of a Survey", Proc. of the ACM International Symposium on Wikis (Wikisym 2006), Odense, Denmark, 2006.

[38] Mizoguchi R, Ikeda M, Sinitsa K, "Roles of Shared Ontology in AI-ED Research, Intelligence, Conceptualization, Standardization, and Reusability", Proc. of the 8th World Conference on Artificial Intelligence in Education, 1997, IOS Press.

[39] Muljadi H. and Takeda H."Semantic Wiki as an Integrated Content and Metadata Management System", Proc. of the 4th International Semantic Web Conference,, Galway, Ireland, 2005, , LNCS 3729, Springer Verlag.

[40] Newman R., "Tag Ontology Design", 2007, http://www.holygoat.co.uk/blog/entry/2005-03-23-2

[41] Olsen H., "Navigation blindness, how to deal with the fact that people tend to ignore navigation tools", The Interaction Designer's Coffee Break, Issue 13, Q1 2005.

[42] Passant A., "Using Ontologies to Strengthen Folksonomies and Enrich Information Retrieval in Weblogs: Theoretical Background and Corporate Use-Case", Proc. of the International Conference ICWSM, Boulder, USA, 2007

[43] Powers S., "Cheap Eats at the Semantic Web Café", 2005, http://weblog.burningbird.net/archives/2005/01/27/cheap-eats-at-the-semantic-web-cafe/.

[44] Prud'hommeaux E., Seaborne A., SPARQL Query Language for RDF, W3C Candidate Recommendation 14 June 2007, http://www.w3.org/TR/rdf-sparql-query/

[45] RDF Wiki: http://infomesh.net/2001/05/sw/#rdfwiki,

[46] Schaffert S., Gruber A., and Westenthaler R., "A Semantic Wiki for Collaborative Knowledge Formation.", Proc. of the Semantics 2005, Vienna, Austria.

[47] Schaffert S., Bischof D., Buerger T., Gruber A., Hilzensauer W. and Shaffert S, "Learning with Semantic Wikis", Proc of 1st Workshop on Semantic Wikis, Budva, Montenegro, 2006.

[48] Smith G., "IA Summit Folksonomies Panel". 2005, http://atomiq.org/archives/2005/03/ia_summit_folksonomies_panel.html.

[49] Souzis A., "Building a Semantic Wiki", IEEE Intelligent Systems, 20(5), September/October, 2005

[50] Völkel M., Oren E.,"Towards a Wiki Interchange Format (WIF)", Proc. of the 1st Workshop on Semantic Wikis, Budva, Montenegro, 2006.

[51] Völkel M., Krötzsch M., Vrandečić D., Haller H., Studer R, "Semantic wikipedia", Proc. of the 15th World Wide Web Conference. Edinburgh, Scotland, 2006, ACM Press.

[52] Vrandecic D., Krötzsch M., "Reusing Ontological Background Knowledge in Semantic Wikis", Proc. of the 1st Workshop on Semantic Wikis, Budva, Montenegro, 2006..

[53] WikiOnt: http://boole.cs.iastate.edu:9090/wikiont/,

[54] W3C, Semantic Web Activity, http://www.w3.org/2001/sw/ and http://www.w3.org/2001/sw/Activity